Knowledge representation

lecture 4

Html annotations & rdfa + rdfs & rules and logic

Tanel Tammet

TTU

Lecture overview

- Html annotations:
  - RDFa
  - Microformats
  - Facebook & Google
- Schema.org
- RDFs
- RDFs and logic

Semantic html annotations

Big question: how to publish machine-processable data (as opposed to visual-oriented html) on the web

Two main approaches:

- Publish data in csv, xml, json, rdf or some such purely machine-oriented format
- Publish data inside human/visual-oriented html

Semantic html annotations: integrate machine-processable information into visual-design-oriented html

Semantic html „standards“

Important formats:

- **RDFa**: W3C standard pushed by the semantic web community
  - Facebook Open Graph protocol
  - Google Structured data

- **Microdata**
- **Microformats**

Important ontologies/dictionaries:

- **schema.org**
- **wordnet**

W3C standard: RDFa

Read:

http://en.wikipedia.org/wiki/RDFa
http://www.w3.org/TR/xhtml-rdfa-primer/

History:

2008: RDFa 1.0 reached W3C Recommendation status
2012: RDFa 1.1 does not require XML-specific namespace mechanism

Näide 1:

Algse html-teksti

```
<tr>
   <td>Jaanus Kask</td>
   <td>6024554</td>
   <td><a href=”http://kask.googlepages.com”>kliki siia</a></td>
</tr>
```

asemel on html-tekstiks RDFa järgi annoteeritult

```
<tr about=”#jaanus_kask”>
   <td property=”er:nimi”>Jaanus Kask</td>
   <td property=”er:telefon”>6024554</td>
   <td><a rel=”er:koduleht”
            href=”http://kask.googlepages.com”>kliki siia</a></td>
</tr>
```

Näide 2:

Algse html teksti

```
<div>
<a class="tootaja_nimi"
href="http://www.mkm.ee/index.php?id=7187&amp;tootaja=10000450">
Taivo Kivistik</a></div>
<div class="tootaja_inf">
asekantsler (õigusala)
<br>Tel:  6256346 | E-post:
<a class="kontakt_mail" href="javascript:void(0)"
onclick="sendmail('taivo.kivistik', 'mkm.ee')"
>taivo.kivistik
<img src="majandus_juhtkond_files/e_post.gif"
class="e_post_gif" border="0">mkm.ee</a> <br></div><br>
```

Näide 2:

asemel kasutatakse RDFa järgi annoteeritud htmli

```
<span about="#Taivo_Kivistik">
      <div>
      <a class="tootaja_nimi" rel="er:koduleht"
href="http://www.mkm.ee/index.php?id=7187&amp;tootaja=10000450">
      <span property="er:nimi">Taivo Kivistik</span></a></div>
      <div class="tootaja_inf">
      <span property="er:amet">asekantsler (õigusala)</span>
      <br>Tel:  <span property="er:telefon">6256346</span> | E-post:
      <a class="kontakt_mail" href="javascript:void(0)"
      onclick="sendmail('taivo.kivistik', 'mkm.ee')" property="er:email"
      content="taivo.kivistik@mkm.ee">taivo.kivistik
      <img src="majandus_juhtkond_files/e_post.gif"
      class="e_post_gif" border="0">mkm.ee</a> <br></div><br>
</span>
```

RDFa konkreetsemalt:

Üldiselt peaks piisama neist lihtsamatest RDFa võimalustest:

objekti id määramiseks:	**about**=”#kohalik_id”
lingi omaduse nime jaoks:	**rel**=”er:omadus”
üldiselt omaduse nime jaoks:	**property**=”er:omadus”
kui vaja (enamasti ei), tüüp:	**type**=”xsd:tyybinimi”
omaduse väärtus:	tüüpiliselt htmls olemas
omaduse väärtus alternatiivselt:	**content**=”väärtus”

Facebook & RDFa: Open Graph protocol

https://developers.facebook.com/docs/opengraphprotocol/

```
<meta content="Sightsmap" property="og:title">
<meta content="http://www.sightsmap.com" property="og:url">
<meta content="Sightsmap" property="og:site_name">
<meta content="website" property="og:type">
<meta content="Sightseeing popularity heatmaps for the whole world, based
    on Panoramio photos, Wikipedia and FourSquare."
    property="og:description">
<meta content="http://www.sightsmap.com/wpng/siteimage_small_150.jpg"
    property="og:image">
<meta content="tanel.tammet" property="fb:admins">
<meta content="330325837036787" property="fb:app_id">
```

Google & RDFa:
Rich snippets

http://support.google.com/webmasters/bin/topic.py?hl=en&topic=1088472&parent=21997&ctx=topic

Use either:

- RDFa
- Microdata
- Microformats

Microdata:

```
<body itemtype="http://schema.org/WebPage" itemscope="">
```

Or a longer example:

```
<div itemscope itemtype="http://data-vocabulary.org/Person">
  My name is <span itemprop="name">Bob Smith</span>
  but people call me <span itemprop="nickname">Smithy</span>.
  Here is my home page:
  <a href="http://www.example.com" itemprop="url">www.example.com</a>
  I live in Albuquerque, NM and work as an <span itemprop="title">engineer</span>
  at <span itemprop="affiliation">ACME Corp</span>.
</div>
```

Compare RDFa & microdata

RDFa:

```
<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Person">
  My name is <span property="v:name">Bob Smith</span>,
  but people call me <span property="v:nickname">Smithy</span>.
  Here is my homepage:
  <a href="http://www.example.com" rel="v:url">www.example.com</a>.
  I live in Albuquerque, NM and work as an <span property="v:title">engineer</span>
  at <span property="v:affiliation">ACME Corp</span>.
</div>
```

Microdata:

```
<div itemscope itemtype="http://data-vocabulary.org/Person">
  My name is <span itemprop="name">Bob Smith</span>
  but people call me <span itemprop="nickname">Smithy</span>.
  Here is my home page:
  <a href="http://www.example.com" itemprop="url">www.example.com</a>
  I live in Albuquerque, NM and work as an <span itemprop="title">engineer</span>
  at <span itemprop="affiliation">ACME Corp</span>.
</div>
```

Schema.org

Created by: Google, Microsoft, and Yahoo

Goal: create a shared **markup vocabulary** supported by major search engines

See also: http://www.sitemaps.org/

Compare to: wordnet (a very different animal)

See additionally:

http://googlewebmastercentral.blogspot.com/2012/12/introducing-data-highlighter-for-event.html
http://googlewebmastercentral.blogspot.com/2012/07/introducing-structured-data-dashboard.html

RDFS

RDFS: RDF Schema

Three kinds of simple taxonomy rules added to RDF

„if X is a car, X is a vehicle“

„if X has a property profession, X is a person“

„if X has a property brother, value of the property is a person“

First (main) rule

example:

ex:MotorVehicle rdf:type rdfs:Class

exthings:companyCar rdf:type ex:Van

ex:Van rdfs:subClassOf ex:MotorVehicle

Main rule

same facts in the predicate calculus notation:

rdf:type(ex:MotorVehicle, rdfs:Class)

rdf:type(exthings:companyCar,ex:Van)

rdfs:subClassOf(ex:Van, ex:MotorVehicle)

built-in rule assumed in rdfs:

rdf:type(X,Y) & rdfs:subClassOf(Y,Z) => rdf:type(X,Z)

Second rule

ex:Person rdf:type rdfs:Class .
ex:author rdf:type rdf:Property .
ex:author rdfs:range ex:Person

and the fact

ex:person1 ex:author ex:hamlet

should derive:

ex:author rdf:type ex:Person

Logically ...

built-in rule assumed in rdfs:

Y(X,Z) & rdfs:range(Y,U) => rdf:type(X,U)

Third rule

ex:Book rdf:type rdfs:Class .
ex:author rdf:type rdf:Property .
ex:author rdfs:domain ex:Book .

and the fact

ex:person1 ex:author ex:hamlet

should derive:

ex:hamlet rdf:type ex:Book

Logically ...

built-in rule assumed in rdfs:

Y(X,Z) & rdfs:domain(Y,U) => rdf:type(Z,U)

Additional encoding layer!

built-in rule assumed in rdfs:

rdf:type(X,Y) & rdfs:subClassOf(Y,Z) => rdf:type(X,Z)
Y(X,Z) & rdfs:range(Y,U) => rdf:type(X,U)
Y(X,Z) & rdfs:domain(Y,U) => rdf:type(Z,U)

have to be encoded in classical 1st order logic as

rdf(X,rdf:type,Y) & rdf(Y,rdfs:subClassOf,Z) => rdf(X,rdf:type,Z)
rdf(X,Y,Z) & rdf(Y,rdfs:range,U) => rdf(X,rdf:type,U)
rdf(X,Y,Z) & rdf(Y,rdfs:domain,U) => rdf(Z,rdf:type,U)

NOT provided in rdfs:

- cardinality constraints on properties, e.g., that a Person has exactly one biological father.
- specifying that a given property (such as ex:hasAncestor) is transitive, e.g., that if A ex:hasAncestor B, and B ex:hasAncestor C, then

  A ex:hasAncestor C.
- specifying that a given property is a unique identifier (or key) for instances of a particular class.
- specifying that two different classes (having different URIrefs) actually represent the same class.
- specifying that two different instances (having different URIrefs) actually represent the same individual.
- specifying constraints on the range or cardinality of a property that depend on the class of resource to which a property is applied, e.g., being able to say that for a soccer team the ex:hasPlayers property has 11 values, while for a basketball team the same property should have only 5 values.
- the ability to describe new classes in terms of combinations (e.g., unions and intersections) of other classes, or to say that two classes are disjoint (i.e., that no resource is an instance of both classes).

Ways to extend rdfs

Two approaches:

- Using traditional rules (current mainstream: RIF)
- Using a specialised description-logic based language OWL (Web Ontology Language)